

# Web 安全第二次作业：MD5 算法程序设计与实现

陈铭涛

16340024

## 算法原理概述

MD5 算法是一种广泛使用的 Hash 算法，常用于确保信息传输的完整性与一致性。对于输入的任一不定长度信息，MD5 算法在对最后一块进行填充后用 512 个比特对其进行分组，使用压缩函数将其生成 4 个 32 比特的数据，合并为 128 比特的信息摘要。MD5 实现的基本过程为填充，分块，缓冲区初始化，循环压缩，得出结果。

## 算法总体流程

1. Padding: 对于给出的长度为  $K$  比特的数据, MD5 将其填充  $P$  比特的数据, 数据为  $100\dots0$ , 填充后  $K+P$  模 512 后为 448。填充的数据量范围为 1 至 512 位比特。完成上步后在数据再填充  $K$  按 little endian 排列的 64 位数据, 最终得到的数据长度  $K+P+64$  模 512 后为 0.

2. 初始化: 初始化 32 比特寄存器  $A$ ,  $B$ ,  $C$ ,  $D$  作为初始向量, 数据如下:

$A=0x67452301$

$B=0xEFCDAB89$

$C=0x98BADCFE$

$D=0x10325476$

要注意的是需要按 little endian 排列字节数据，即低位字节放在内存的低地址位置。

3. 压缩函数：将消息按 512 比特分组为单位与 CV 向量传入压缩函数，压缩函数进行 4 轮 16 次迭代的循环输出 128 位下一组的 CV 值，最后一组数据输出的 CV 值即为 MD5 的结果。
4. 循环迭代：压缩函数输入的 128 位 CV 值划分为各 32 位的 a, b, c, d 值进行循环迭代，每轮循环各使用如下的对应生成函数 g 结合对应的 T 表元素与消息的不同部分 X 进行 16 次迭代运算：

第一轮循环：  $F(b, c, d) = (b \wedge c) \vee (\neg b \wedge d)$

第二轮循环：  $G(b, c, d) = (b \wedge d) \vee (c \wedge \neg d)$

第三轮循环：  $H(b, c, d) = b \oplus c \oplus d$

第四轮循环：  $I(b, c, d) = c \oplus (b \vee \neg d)$

每一轮的迭代运算为  $a = b + ((a + g(a, b, c) + X[k] + T[i]) \lll s)$

其中  $X[k]$  表示消息组中的第 k 个 32 位字， $T[i]$  表示 T 表的第 i 个元素，

$\lll s$  表示将数据循环左移 s 位，位移的量来自于 s 值表。

运算后将 b, c, d, a 分别轮换为 a, b, c, d。

最后一个分组完成循环迭代后的结果即为 MD5 输出的结果。

5. 数据表：上面使用的  $X[k]$  中 k 的确定方法为：对于每轮的第 i 次迭代 ( $i=1..16$ )，令  $j=i-1$ ，则对于第一轮迭代  $k=j$ ，第二轮迭代  $k=(1+5j) \bmod 16$ ，第三轮迭代  $k=(5+3j) \bmod 16$ ，第四轮迭代  $k=7j \bmod 16$ 。

T 表的生成方式为：对于 T 表中的第 i 个元素， $T[i] = \text{int}(2^{32} \times |\sin(i)|)$ 。

循环左移的 s 值表如下：

```
s[1..16] = { 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22 }  
s[17..32] = { 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20 }  
s[33..48] = { 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23 }  
s[49..64] = { 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21 }
```

## 代码实现简述

提交的代码使用 C++ 语言编写，包含的文件分别为：

- md5.cpp, md5.h 分别为 MD5 类的实现与定义，为程序的主要部分
- md5test.cpp, md5test.h 为进行测试的函数的实现与定义，用于测试与辅助显示 md5 结果。
- main.cpp 为程序主函数，其使用方法参考了 macOS 系统终端自带的 md5 命令使用方法。
- MD5 类为主要的內容，其包含的各方法与介绍如下：

```
std::array<unsigned char, 16> sum(const std::vector<unsigned char> &data);
```

对于输入的字节 vector 进行 md5 计算，返回 16 字节（128 比特）的 md5 结果。

```
std::array<unsigned char, 16> sum(const std::string &data);
```

对传入的字符串进行 md5 计算，返回 16 字节（128 比特）的 md5 结果。

```
std::array<unsigned char, 16> md5Sum(std::vector<unsigned char> data)
```

上面两个函数实际调用的函数，包含了 md5 算法的全过程。

```
void padding(std::vector<unsigned char> & data);
```

对传入的消息数据根据算法的方法原地进行填充操作。

```
uint32_t circularLeftShift(uint32_t data, unsigned int c);
```

对传入的 32 位数据进行循环左移 c 位

```
std::array<unsigned char, 16> HMD5(const std::array<unsigned char, 64> &data, const std::array<unsigned char, 16> &cv);
```

压缩函数，传入数据与 cv 值，其中包含了 4 轮 16 次循环迭代操作，返回 128 位运算后的结果。

```
auto F = [](uint32_t b, uint32_t c, uint32_t d) { return (b & c) | (~b & d); };  
auto G = [](uint32_t b, uint32_t c, uint32_t d) { return (b & d) | (c & ~d); };  
auto H = [](uint32_t b, uint32_t c, uint32_t d) { return b ^ c ^ d; };  
auto I = [](uint32_t b, uint32_t c, uint32_t d) { return c ^ (b | ~d); };  
array<function<uint32_t(uint32_t, uint32_t, uint32_t)>, 4> g = {F, G, H, I};
```

该 4 个函数为 HMD5 函数实现中使用的 4 个轮函数 g。

- md5test 中包含的函数介绍如下：

```
void md5StrTest(const std::string& input)
```

将传入的字符串进行 md5 运算，结果以 16 进制形式打印至标准输出

```
void md5FileTest(const char* filename)
```

传入文件名，使用二进制模式读取文件并进行 md5 运算，结果以 16 进制输出。

```
void md5RunTest();
```

使用预先设定好的一组字符串进行 md5 运算测试，输出结果。

```
std::string bytesToHexStr(std::array<unsigned char, 16> data);
```

将输入的 128 比特数据转为 16 进制形式的字符串。

## 程序编译

程序使用 `cmake` 进行编译，在 Windows, Mac 和 Linux 系统下编译指令分别如下：

```
# Linux, macOS
cmake .
make

#Windows
cmake -G "MinGW Makefiles" .
mingw32-make
```

编译完成后将会把可执行文件输出至 `bin` 文件夹下，提交的代码目录下的 `bin` 文件夹已包含这三个系统下的可执行文件。

程序的使用方法模仿了 macOS 命令行下的 `md5` 工具，具体如下：

```
./md5 [-x] [-s string] [files ...]
```

其中各参数作用如下：

- `-x` 会使程序执行测试，使用预先定义的一组字符串进行 `md5` 运算结果的测试。
- `-s` 后接字符串，程序会对该字符串进行运算并输出结果，该参数可重复指定
- 程序读取到第一个非选项的参数时将会进入文件读取模式，将剩余的参数视为文件名并读取文件进行运算输出结果。
- 若未指定任何参数，程序将会从标准输入中读取输入，并在用户输入 `EOF` 后输出结果。

## 程序运行结果

在 macOS 10.14.1 系统下进行程序的运行测试，与系统自带 md5 工具进行结果

对比，程序运行的效果如下：

1. 从标准输入中读取数据：

```
3. mig@ai: ~/Desktop/大三上/web-security/hw/infosec/md5/md5/bin (zsh)
~/Desktop/大三上/web-security/hw/infosec/md5/md5/bin master ./md5-mac
Can this work?
So?
21c7c6982c570bde4884eb99e8588cd2
~/Desktop/大三上/web-security/hw/infosec/md5/md5/bin master md5
Can this work?
So?
21c7c6982c570bde4884eb99e8588cd2
~/Desktop/大三上/web-security/hw/infosec/md5/md5/bin master
```

2. 对代码文件以及可执行文件自身计算 md5 值：

```
3. mig@ai: ~/Desktop/大三上/web-security/hw/infosec/md5/md5/bin (zsh)
~/Desktop/大三上/web-security/hw/infosec/md5/md5/bin master ./md5-mac ../md5.cpp
MD5 (../md5.cpp) = f67e2567092b39660a876419de0ea669
~/Desktop/大三上/web-security/hw/infosec/md5/md5/bin master md5 ../md5.cpp
MD5 (../md5.cpp) = f67e2567092b39660a876419de0ea669
~/Desktop/大三上/web-security/hw/infosec/md5/md5/bin master ./md5-mac md5-mac
MD5 (md5-mac) = 8b8c6d4804202620b8577aafc7c7f5fb
~/Desktop/大三上/web-security/hw/infosec/md5/md5/bin master md5 md5-mac
MD5 (md5-mac) = 8b8c6d4804202620b8577aafc7c7f5fb
~/Desktop/大三上/web-security/hw/infosec/md5/md5/bin master
```

3. 从命令行读取字符串：

```
3. mig@ai: ~/Desktop/大三上/web-security/hw/infosec/md5/md5/bin (zsh)
~/Desktop/大三上/web-security/hw/infosec/md5/md5/bin master ./md5-mac -s Hi....
MD5 ('Hi....') = 7f3342c10046676568672ce7b283c2c7
~/Desktop/大三上/web-security/hw/infosec/md5/md5/bin master md5 -s Hi....
MD5 ("Hi....") = 7f3342c10046676568672ce7b283c2c7
~/Desktop/大三上/web-security/hw/infosec/md5/md5/bin master ./md5-mac -s ok -s yes
MD5 ('ok') = 444bcb3a3fcf8389296c49467f27e1d6
MD5 ('yes') = a6105c0a611b41b08f1209506350279e
~/Desktop/大三上/web-security/hw/infosec/md5/md5/bin master md5 -s ok -s yes
MD5 ("ok") = 444bcb3a3fcf8389296c49467f27e1d6
MD5 ("yes") = a6105c0a611b41b08f1209506350279e
~/Desktop/大三上/web-security/hw/infosec/md5/md5/bin master
```

4. 使用-x 执行测试：

```
3. mig@ai: ~/Desktop/大三上/web-security/hw/infosec/md5/md5/bin (zsh)
~/Desktop/大三上/web-security/hw/infosec/md5/md5/bin master ./md5-mac -x
MD5 test suite:
MD5 ('') = d41d8cd98f00b204e9800998ecf8427e - verified correct
MD5 ('a') = 0cc175b9c0f1b6a831c399e269772661 - verified correct
MD5 ('abc') = 900150983cd24fb0d6963f7d28e17f72 - verified correct
MD5 ('message digest') = f96b697d7cb7938d525a2f31aaf161d0 - verified correct
MD5 ('abcdefghijklmnopqrstuvwxyz') = c3fcd3d76192e4007dfb496cca67e13b - verified correct
MD5 ('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789') = d174ab98d277d9f5a5611c2c9f419d9f - verified correct
MD5 ('12345678901234567890123456789012345678901234567890123456789012345678901234567890') = 57edf4a22be3c955ac49da2e2107b67a - verified correct
MD5 ('MD5 has not yet (2001-09-03) been broken, but sufficient attacks have been made that its security is in some doubt') = b50663f41d44d92171cb9976bc118538 - verified correct
~/Desktop/大三上/web-security/hw/infosec/md5/md5/bin master md5 -x
MD5 test suite:
MD5 ('') = d41d8cd98f00b204e9800998ecf8427e - verified correct
MD5 ('a') = 0cc175b9c0f1b6a831c399e269772661 - verified correct
MD5 ('abc') = 900150983cd24fb0d6963f7d28e17f72 - verified correct
MD5 ('message digest') = f96b697d7cb7938d525a2f31aaf161d0 - verified correct
MD5 ('abcdefghijklmnopqrstuvwxyz') = c3fcd3d76192e4007dfb496cca67e13b - verified correct
MD5 ('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789') = d174ab98d277d9f5a5611c2c9f419d9f - verified correct
MD5 ('12345678901234567890123456789012345678901234567890123456789012345678901234567890') = 57edf4a22be3c955ac49da2e2107b67a - verified correct
MD5 ('MD5 has not yet (2001-09-03) been broken, but sufficient attacks have been made that its security is in some doubt') = b50663f41d44d92171cb9976bc118538 - verified correct
~/Desktop/大三上/web-security/hw/infosec/md5/md5/bin master
```

程序在 macOS 10.14.1, Ubuntu 18.04.1 LTS 以及 Windows 10 下测试运行正常，输出结果与各系统自带的 md5 工具输出的结果相同。